2013

Enhanced ant colony algorithm for cost-aware data-intensive service provision

Lijuan Wang
University of Wollongong, lw840@uowmail.edu.au

Jun Shen
University of Wollongong, jshen@uow.edu.au

Ghassan Beydoun
University of Wollongong, beydoun@uow.edu.au

Publication Details

Wang, L., Shen, J. & Beydoun, G. (2013). Enhanced ant colony algorithm for cost-aware data-intensive service provision. 2013 IEEE International Congress on Big Data: BigData Congress (pp. 227-234). United States: IEEE.

Enhanced ant colony algorithm for cost-aware data-intensive service provision

Abstract

Huge collections of data have been created in recent years. Cloud computing has been widely accepted as the nextgeneration solution to addressing data-proliferation problems. Because of the explosion in digital data and the distributed nature of the cloud, as well as the increasingly large number of providers in the market, providing efficient cost models for composing dataintensive services will become central to this dynamic market. The location of users, service composers, service providers, and data providers will affect the total cost of service provision. Different providers will need to make decisions about how to price and pay for resources. Each of them wants to maximize its profit as well as retain its position in the marketplace. Based on our earlier work, this paper addresses the effect of data intensity and the communication cost of mass data transfer on service composition, and proposes a service selection algorithm based on an enhanced ant colony system for data-intensive service provision. In this paper, the data-intensive service composition problem is modeled as an AND/OR graph, which is not only able to deal with sequence relations and switch relations, but is also able to deal with parallel relations between services. In addition, the performance of the service selection algorithm is evaluated by simulations.

Keywords

provision, ant, enhanced, intensive, data, aware, cost, algorithm, colony, service

Publication Details

Wang, L., Shen, J. & Beydoun, G. (2013). Enhanced ant colony algorithm for cost-aware data-intensive service provision. 2013 IEEE International Congress on Big Data: BigData Congress (pp. 227-234). United States: IEEE.

Enhanced ant colony algorithm for cost-aware data-intensive service provision

Lijuan Wang, Jun Shen, Ghassan Beydoun
School of Information Systems and Technology
University of Wollongong
NSW, Australia
Email: lw840@uowmail.edu.au, jshen@uow.edu.au, beydoun@uow.edu.au

Abstract—**Huge collections of data have been created in recent years. Cloud computing has been widely accepted as the next-generation solution to addressing data-proliferation problems. Because of the explosion in digital data and the distributed nature of the cloud, as well as the increasingly large number of providers in the market, providing efficient cost models for composing data-intensive services will become central to this dynamic market. The location of users, service composers, service providers, and data providers will affect the total cost of service provision. Different providers will need to make decisions about how to price and pay for resources. Each of them wants to maximize its profit as well as retain its position in the marketplace. Based on our earlier work, this paper addresses the effect of data intensity and the communication cost of mass data transfer on service composition, and proposes a service selection algorithm based on an enhanced ant colony system for data-intensive service provision. In this paper, the data-intensive service composition problem is modeled as an AND/OR graph, which is not only able to deal with sequence relations and switch relations, but is also able to deal with parallel relations between services. In addition, the performance of the service selection algorithm is evaluated by simulations.**

Keywords—***ant colony system algorithm, data-intensive service composition, quality of service.***

I. Introduction

In recent years, huge collections of data have been created by the advances in technology areas such as digital sensors, communications, computation, and storage. This data deluge exhibits not just volume and velocity but also variability and diversity of structure, completeness, and domain [33]. The impact of enormous new sources of data extends to many areas of society, far beyond business, industry, government, science, sports, advertising and public health, with no area being untouched. There is no doubt that the importance of data-intensive computing has been increasing and it becomes the foremost research field in industry and academic communities. As a result, applications based on data-intensive services have become the most challenging type of applications in service-oriented computing. Generally, we define the notion of data-intensive services as Web services that make use of very large data sets as inputs. Services use data from data providers, and they also use data from other services. The data from other services sometimes consist of that which is exchanged between services within a common context. In this paper, we focus on the data from data providers as this kind of data is generally much larger.

To solve a complex social or scientific problem such as, for example, attempting to understand the regional scale impacts of fires, or the long-range transportation of pollutants on air quality, or even the implications of climate change, scientists need to combine data from aircraft and satellites. And it is necessary to design a workflow of various data-intensive services and get a composite data-intensive service when using Web service technologies to solve such problems. A composite service can be re-used by other users. The research area of service composition has been attracting tremendous attention in recent years [1], [5], [10], [13], [23], [27], [32], however, very little research has considered the effect of data intensity and the communication cost of mass data transfer to service composition, and most research has considered only the sequence relation between services. On the other hand, it should be noted that the optimization processes in most QoS-aware service composition studies aim to satisfy quality constraints and provide the “best” solutions with highest QoS. However, from a cost perspective, consumers will not always pay more, even if the qualities of the requested composite services exceed their expectations. This is because most service composition processes involve static-pricing models. In addition, data-intensive services need to access large data sets that may each be replicated in different data centers [12], [17], [25], but the access cost of each data replica on one data center is different from that for other data centers [19]. Therefore, how to select appropriate data centers for accessing data replicas and how to select services with the lowest associated costs are emerging problems when deploying and executing data-intensive service applications.

In a data-intensive service composition process, a service composer, service providers, and data providers all want to have a position in the market whilst maximizing their profits. Data play the dominant role in the process of data-intensive service composition. The authors of [6] were concerned with data placement policies for application execution. The paper [3] focused primarily on the problem of optimizing the replication of data, that is, deciding when and where to create and delete replicas of data files. We aim to minimize the overall cost of data-intensive service composition and maximize the three actors’ (service composer, service providers, data providers) market competitiveness in the “long-term”. Traditional quality-driven composition techniques usually consider the qualities at the time of the composition [11]. It is not easy for service providers to compete on price if they want to establish long-term partnerships in the marketplace, since they are eager to have an acceptable profit when they sell their services. If they cannot have position in the marketplace, the problem may be that their costs are too high rather than that the price is too low. So, during the process of data-intensive service composition, it is necessary to adopt dynamic pricing models and all actors in the value chain negotiate with each other to lower cost.

In our data-intensive service composition model, there are two parts. The first part is that the service composer selects service candidates while the service providers select data replicas, and the second part is that the service composer negotiates with multiple service providers while the service providers negotiate with their respective data providers at the same time. The service composer wants to get a satisfactory service according to QoS requirements in the first negotiation process, and the service providers want to lower the price of data in the second negotiation process. We have already done pilot studies in applying bio-inspired algorithms to tackle this kind of problem [14], [15], [20], [21], [30]. Based on our earlier outcomes and other studies, we will design a holistic cost minimization data-intensive service composition mechanism. The solution of the first part is presented in this paper. An enhanced service selection algorithm based on an ant colony system for cost minimized data-intensive service composition is proposed.

The remainder of this paper is organized as follows: Section II introduces the background. Section III details the utility function for local optimization and global optimization, and it also presents a local selection approach as well as the problem statement. Section IV investigates how an ant colony algorithm could be used to solve data-intensive service composition problems. Section V shows the experiments and analysis. Section VI reviews some related work. Finally, Section VII concludes this paper and proposes future work.

II. Background

A. Service composition as a workflow design problem

According to [4], a composite service is similar to a workflow. The authors of [2] proposed a model for service composition using a workflow. They argued that processes could be viewed as workflows that manage services instead of tasks. Thus, in a workflow-based service composition model, a workflow is actually a composite service. The types of workflow structures in this paper were derived from the workflow patterns in [18]. The basic workflow structures are: *sequence, AND split (fork), XOR split (switch), Loop, AND join*, and *XOR join (merge)*. Accordingly, the control flows between services are shown in Fig. 1. The irregular circle represents an abstract service.

B. Service composition model using an AND/OR graph

In the field of service composition, a directed graph is used to represent the dependencies between services. This is referred to as 'service functional graph'. Discovering a composite service is essentially searching for a solution in a solution space represented by the 'service candidate graph'. Fig. 2 is a 'service candidate graph', and Fig. 3 is a 'service functional graph' in which data sets, as the inputs of services, are incorporated.

As shown in Fig. 3, there is a logical AND relationship between AS_2 and AS_3, and there is a logical OR relationship

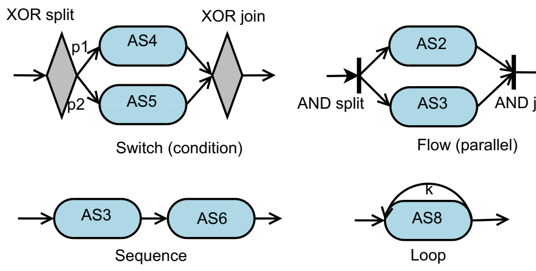


Fig. 1: Control structures between services

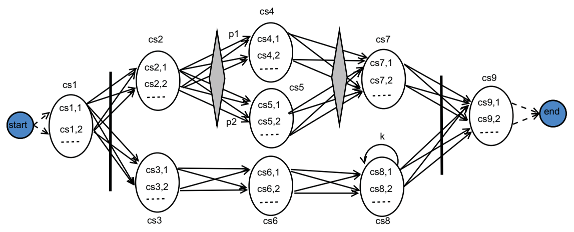


Fig. 2: Service Candidate Graph

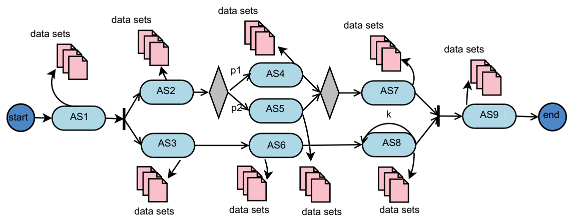


Fig. 3: Service functional graph for data-intensive service composition

between AS_4 and AS_5. Thus, an AND/OR graph is used to form the service composition model. An AND/OR graph is a 'service functional graph', in which a vertex is called an 'AND split' vertex if there is a logical AND relationship between its direct successors, and a vertex is called an 'OR split' vertex if there is a logical OR relationship between its direct successors. Meanwhile, a vertex is called an 'AND join' vertex if there is a logical AND relationship between its direct predecessors, and a vertex is called an 'OR join' vertex if there is a logical OR relationship between its direct predecessors. All 'AND split' and 'AND join' vertices are called AND vertices, and all 'OR split' and 'OR join' vertices are called OR vertices. Each 'AND split' vertex and its 'AND join' vertex are a pair, and each 'OR split' vertex and its 'OR join' vertex are a pair. For example, AS_1 and AS_9 are AND vertices, AS_2 and AS_7 are OR vertices. In this paper, we just consider a graph with basic control structures as shown in Fig. 1.

III. Data-intensive service composition

A. Utility function

In order to evaluate the quality of a concrete service, a utility function is used. This paper adopts the simple additive weighting (SAW) approach in the multiple criteria decision making (MCDM) [28] technique for the utility function. The utility computation includes two phases: the scaling phase and the weighting phase. The scaling phase is used to normalize all QoS attributes to the same scale, independent of their units and ranges. The weighting phase is used to compute the overall utility for each service candidate by using weights depending on users' priorities and preferences. In this section, two utility functions will be presented, namely, utility function for a concrete service, and utility function for a composite service.

1) Select a concrete service using the local utility function: Suppose a composite service CS is composed of n tasks, and there are m concrete services to execute each task. Each concrete service $cs_{i,j}$ is associated with a QoS vector $q_{ij} = [q_{ij}^1, q_{ij}^2, \ldots, q_{ij}^r]$ with r QoS parameters. $Q_{k,i}^{max} = \max\limits_{\forall cs_{i,j} \in cs_i} q_{ij}^k$ and $Q_{k,i}^{min} = \min\limits_{\forall cs_{i,j} \in cs_i} q_{ij}^k$ are used to represent the maximal value and the minimal value of the k-th QoS attributes of all concrete services in candidate set cs_i respectively. The set of QoS attributes can be classified into two groups: positive and negative QoS attributes. The values of negative QoS attributes like response time need to be minimized. The higher their values, the lower the QoS. The values of positive QoS attributes such as availability need to be maximized. The higher their values, the higher the QoS. Suppose there are α negative QoS attributes and β positive QoS attributes in the QoS vector of concrete service cs_{ij}, and $\alpha + \beta = r$. Now the utility of $cs_{i,j}$ is computed according to (1).

$$\begin{aligned} U(cs_{i,j}) = & \sum_{k_1=1}^{\alpha} \left(\left(\frac{Q_{k_1,i}^{max} - q_{ij}^{k_1}}{Q_{k_1,i}^{max} - Q_{k_1,i}^{min}} \right) * W_{k_1} \right) \\ & + \sum_{k_2=1}^{\beta} \left(\left(\frac{q_{ij}^{k_2} - Q_{k_2,i}^{min}}{Q_{k_2,i}^{max} - Q_{k_2,i}^{min}} \right) * W_{k_2} \right) \\ & \frac{Q_{k_1,i}^{max} - q_{ij}^{k_1}}{Q_{k_1,i}^{max} - Q_{k_1,i}^{min}} = 1, \quad \text{if } Q_{k_1,i}^{max} - Q_{k_1,i}^{min} = 0 \\ & \frac{q_{ij}^{k_2} - Q_{k_2,i}^{min}}{Q_{k_2,i}^{max} - Q_{k_2,i}^{min}} = 1, \quad \text{if } Q_{k_2,i}^{max} - Q_{k_2,i}^{min} = 0 \end{aligned} \tag{1}$$

where $W_{k_1}, W_{k_2} \in [0, 1]$, and $\sum\limits_{k_1=1}^{\alpha} W_{k_1} + \sum\limits_{k_2=1}^{\beta} W_{k_2} = 1$. W_{k_1} and W_{k_2} represent weights of k_1-th and k_2-th quality criteria with values normally provided by the users based on their own preferences.

2) Select a composite service using the global utility function: Assigning a concrete service to each abstract service in an execution path leads to a possible service composition solution. As in the local utility function approach, a SAW approach is used to select a concrete service. The selection of a composite service also relies on this approach. Q_k^{MAX} and Q_k^{MIN} are used to represent the maximal value and the minimal value of the k-th QoS attributes of all the execution paths respectively.

$$\begin{aligned} Q_k^{MIN} &= \sum_{i=1}^{n} Q_{k,i}^{min} = \sum_{i=1}^{n} \min_{\forall cs_{i,j} \in cs_i} q_{ij}^k \\ Q_k^{MAX} &= \sum_{i=1}^{n} Q_{k,i}^{max} = \sum_{i=1}^{n} \max_{\forall cs_{i,j} \in cs_i} q_{ij}^k \end{aligned} \tag{2}$$

Suppose a composite service CS is associated with a QoS vector $q_{CS} = [q_{CS}^1, q_{CS}^2, \ldots, q_{CS}^r]$. The overall utility of CS is computed according to (3).

$$\begin{aligned} U(CS) = & \sum_{k_1=1}^{\alpha} \left(\left(\frac{Q_{k_1}^{MAX} - q_{CS}^{k_1}}{Q_{k_1}^{MAX} - Q_{k_1}^{MIN}} \right) * W_{k_1} \right) \\ & + \sum_{k_2=1}^{\beta} \left(\left(\frac{q_{CS}^{k_2} - Q_{k_2}^{MIN}}{Q_{k_2}^{MAX} - Q_{k_2}^{MIN}} \right) * W_{k_2} \right) \\ & \frac{Q_{k_1}^{MAX} - q_{CS}^{k_1}}{Q_{k_1}^{MAX} - Q_{k_1}^{MIN}} = 1, \quad \text{if } Q_{k_1}^{MAX} - Q_{k_1}^{MIN} = 0 \\ & \frac{q_{CS}^{k_2} - Q_{k_2}^{MIN}}{Q_{k_2}^{MAX} - Q_{k_2}^{MIN}} = 1, \quad \text{if } Q_{k_2}^{MAX} - Q_{k_2}^{MIN} = 0 \end{aligned} \tag{3}$$

B. Local selection

When performing the local utility function to select a concrete service, each task selects a concrete service without taking into account the other tasks involved in the composite service. Since (1) compares the quality value of a concrete service with the local maximum/minimum value ($Q_{k,i}^{max}$ and $Q_{k,i}^{min}$) of its candidate set, this scaling approach can lead to local optima. In order to make the evaluation of concrete services be valid globally and avoid a local optimum, the utility of a concrete service $cs_{i,j}$ is computed in a way similar to (1) by replacing the local maximum/minimum value with the global maximum/minimum value(Q_k^{MAX} and Q_k^{MIN}).

C. Problem statement

The data-intensive service composition problem is modeled as an AND/OR graph, denoted as $G = (V, E, D, start, end)$, where $V = V_{and} \cup V_{or} \cup V_{other}$ and E represent the vertices and edges of the graph respectively. D represents a set of data sets for services. V_{and} is AND vertices, V_{or} is OR vertices, and V_{other} is the set of other vertices. For simplicity, it is assumed that all data sets used by each service have already been distributed in data centers prior to service composition following a uniform distribution, and we will not deal with the selection of data replicas in this paper. Furthermore, we will consider only the cost and response time of data-intensive services. The QoS of each data-intensive service in (1) to (3) will include the access cost $C_{vi}(DT)$, transfer cost $C_{tr}(DT)$, and transfer time $T_t(DT)$ of data set DT, since they would affect the cost and response time of the service that makes use of DT. The cost and response time of each service candidate $cs_{i,j}$ are given by (4).

$$\begin{aligned} C(cs_{i,j}) &= C_{vi}(DT) + C_{tr}(DT) + C_{sr}(cs_{i,j}) \\ T(cs_{i,j}) &= T_{rp}(cs_{i,j}) + T_t(DT) \end{aligned} \tag{4}$$

where $C_{sr}(cs_{i,j})$ and $T_{rp}(cs_{i,j})$ are the cost and response time of service $cs_{i,j}$ in traditional service composition respectively, DT is the data set as input of service $cs_{i,j}$. Suppose a composite service of Fig. 3 is CS, t_i and c_i are the response time and cost of service $cs_{i,j}$, then the cost and response time of CS are shown by (5).

$$\begin{aligned}Cost(CS) &= c_1 + c_2 + c_3 + p_1 * c_4 + p_2 * c_5 \\ &\quad + c_6 + c_7 + k * c_8 + c_9 \\ Time(CS) &= t_1 + max\big(t_2 + p_1 * t_4 + p_2 * t_5 + t_7, \\ &\quad t_3 + t_6 + k * t_8\big) + t_9\end{aligned} \tag{5}$$

where p_1 and p_2 are the probability of the conditional branches, k is the loop count for *Loop* structure. According to (4), $c_i = C(\sum_{j=1}^{m}(cs_{i,j}x_{ij}))$, $t_i = T(\sum_{j=1}^{m}(cs_{i,j}x_{ij}))$, where x_{ij} is the constraint used to represent only one concrete service is selected to replace each abstract service during the process of service composition. This constraint satisfies $\sum_{j=1}^{m} x_{ij} = 1, x_{ij} \in \{0, 1\}, i \in \{1, \ldots, n\}$, where x_{ij} is set to 1 if concrete service $cs_{i,j}$ is selected to replace abstract service AS_i and 0 otherwise. Note in (5), $n = 9$.

Using an ant colony system algorithm, the problem of finding a data-intensive service composition solution is considered as an optimization problem, in which the overall utility value has to be maximized. Formally, the optimization problem is described as follows. Find a solution CS in graph G by replacing each abstract service AS_i in $V = \{AS_1, AS_2, \ldots, AS_n\}$ with a concrete service $cs_{i,j} \in cs_i$ such that the overall utility $U(CS)$ is maximized.

IV. Dynamic data-intensive service selection based on an ant colony system

The field of 'ant colony algorithms' studies models derived from the behavior of real ants and it is widely used for combinatorial optimization problems. Ant colony algorithms are developed as heuristic methods to identify efficient paths through a graph and have been applied to identify optimal solutions for service composition problems. The features of ant colony algorithms include positive feedback and local heuristics. An ant colony optimization (ACO) algorithm iteratively performs a loop containing two basic procedures. The first is how the ants construct solutions to the problem, and the second is how to update the pheromone trails [16]. Using ACO algorithms to solve combinatorial optimization problems, requires a representation of the problem and the definition of the meaning of pheromone trails, as well as the heuristic information. Because the results of many ACO applications to NP-hard combinatorial optimization problems show that the best performance is achieved when coupling ACO with local optimizers, it is necessary to implement an efficient local search algorithm [8], [16].

The ant colony system (ACS) is an algorithm strongly inspired by the ant system (AS) but it differs from AS in three main aspects [7]. First, the state transition rule provides a direct way to balance between exploration of new edges and exploitation of *a priori* and accumulated knowledge about the problem. Second, the global updating rule is applied only to edges which belong to the best ant path. Third, a local pheromone updating rule is applied while ants construct a solution.

As discussed, considering different workflow structures and an optimized function with QoS attributes, we model the service composition problem as an AND/OR graph. In the graph, ants are initially positioned on the start vertex. The task of each ant is to find a path from the start vertex to the end vertex. If the ant arrives at an 'AND split' vertex, it will clone several new ants and each ant will choose one of the successors according to the state transition rule respectively. If the ant arrives at an 'OR split' vertex, it only chooses one of the branches with a probability, then will apply the state transition rule to choose the successor. While finding the path, an ant also modifies the amount of pheromone on the edges it has passed by applying the local updating rule. Once all ants arrive at the end vertex, the global updating rule will be applied to modify the amount of pheromone on the edges of the optimal path. The key to ACS for data-intensive service composition problem is how to determine the state transition rule, the global updating rule, the local updating rule, and the ant replication and death rule.

A. State transition rule

When ant k arrives at vertex i, it will choose successor j to move to by applying the rule given by (6).

$$j = \begin{cases} \arg\max_{l \in N_i^k}\{[\tau_{il}][\eta_{il}]^{\beta}\}, & \text{if } q \leq q_0; \\ J, & \text{otherwise.} \end{cases} \tag{6}$$

where q is a random variable uniformly distributed in $[0, 1]$, $q_0 (0 \leq q_0 \leq 1)$ is a parameter, and J is a random variable selected according to the probability distribution given by (7) (with $\alpha = 1$).

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum\limits_{l \in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}}, & \text{if } j \in N_i^k \ ; \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

where p_{ij}^k represents the probability with which ant k, currently at vertex i, chooses to go to vertex j. N_i^k is the neighborhood of vertex i when ant k is in it, that is, the set of vertices that ant k has not visited yet. α is a parameter to control the influence of τ_{ij}, β is a parameter to control the influence of η_{ij}. The neighborhood of vertex i contains all the vertices directly connected to vertex i in the AND/OR graph, except for the predecessor of vertex i. τ_{ij} is the pheromone density on edge (i, j). Here, $\eta_{ij} = U(j)$ which is computed according to the local selection method as heuristic information.

B. Global updating rule

After all ants arrive at the end vertex, a global pheromone updating rule is performed. The pheromone level is updated by applying the global updating rule (8).

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij} \tag{8}$$

where $\rho (0 < \rho < 1)$ is the pheromone evaporation rate, and

$$\Delta\tau_{ij} = \begin{cases} U, & \text{if } \forall(i, j) \in P^{bs} \ ; \\ 0, & \text{otherwise .} \end{cases}$$

where U is the utility of P^{bs} which is computed according to (3). P^{bs} is the best path found since the start of the algorithm. This formula indicates the pheromone trail update, both the evaporation and the new pheromone deposit, only applies to the edges of P^{bs}. There are two types of global updating rule. One is called 'iteration-best' (the best path in the current iteration of the trial), and the other is called 'global-best' (the best path from the beginning of the trial). Experiments have shown that the global-best is slightly better, it is therefore used in our experiment.

C. Local updating rule

When building a solution (i.e., an executed path) of the service composition problem, the ants use a local pheromone update rule that they apply immediately after having crossed edge (i, j) during the path construction which is shown in (9).

$$\tau_{ij} = (1 - \xi)\tau_{ij} + \xi\tau_0, \forall(i, j) \in E. \qquad (9)$$

where $\xi(0 < \xi < 1)$ and τ_0 are two parameters. At the beginning of the search process, a constant amount of pheromone is assigned to all the edges, namely, $\tau_{ij} = \tau_0 = CP$ (CP is a constant, $\forall(i, j) \in E$). The local updating rule will reduce the pheromone trail of edge (i, j) after an ant passes it. In other words, it allows an increase in the exploration of edges that have not been visited yet and, in practice, has the effect that the algorithm does not show stagnation behavior [7].

D. Ant replication and death rule

When the current vertex i of ant k is determined, and i is an 'AND split' vertex which has f direct successors, ant k will make $(f-1)$ copies of itself. Then ant k and the $(f-1)$ ants each will choose one of the direct successor vertices of i to move to, according to the state transition rule described in (6), and different ants choose different successor vertices. During the path construction, each ant uses the local updating rule to the edge that it has passed. Ant k and all of its copies will keep the vertices they have visited in list of ant k. When the $(f-1)$ ants arrive at the 'AND join' vertex of i, all of them will die. The current concrete service at the 'AND join' vertex will be determined by ant k. The whole process is recursive. All the copies of ant k construct paths using the same rule as ant k. That is to say, a copy ant of ant k will make copies of itself when it arrives at an 'AND split' vertex. All copy ants will die when they arrive at the 'AND join' vertex, which pairs with the 'AND split' vertex where they are replicated.

The service selection algorithm based on ACS for data-intensive service composition is given in Algorithm 1. The time complexity of our algorithm is $O(noa * ntask * noc)$, where noa, $ntask$, and noc denote the number of ants, the number of abstract services, and the number of candidate services for each abstract service, respectively.

V. EXPERIMENTS AND ANALYSIS

The proposed approach considers not only sequence relation, but also parallel relation and switch relation between services. In order to evaluate the performance of the proposed algorithm, we verified it against the simulated composite service the functional graph of which is shown in Fig. 3. The loop structure can be unfolded by cloning the vertices involved in the structure as many times as the maximal loop count [31]. This paper does not deal with loop relation so the maximal loop count is 1.

Algorithm 1 Service composition selection algorithm using AND/OR graph based on ACO

Input:
$MaxIt$: the maximum number of iterations;
noa: the number of artificial ants;
$G = (V, E, D, start, end)$: the AND/OR graph;
Output:
S: a service execute path to create a composite service ;

```
1:  $S = \emptyset$; $step = 0$; $\tau_0 = CP$;
2:  while $step < MaxIt$ do
3:    step=step+1;
4:    set all ants at start vertex;
5:    $S_c = \emptyset$;
6:    for each ant $k$ do
7:      $as = \emptyset$;//candidate list for each ant
8:      while ant $k$ is not at the end vertex do
9:        if ant $k$ arrives at an 'AND split' vertex then
10:         ant $k$ executes ant replication and death rule;
11:       else if ant $k$ arrives at an 'OR split' vertex then
12:         ant $k$ chooses only one branch with a probability,
            then chooses a successor according to the state
            transition rule (6);
13:       else
14:         ant $k$ chooses a successor according to the state
            transition rule (6);
15:       end if
16:       update candidate list $as$;
17:       apply the local updating rule (9);
18:     end while
19:     if $U(as) > U(S_c)$ then
20:       $S_c = as$;
21:     end if
22:   end for
23:   when all ants arrive at end vertex, apply global updating
      rule (8) to $S_c$;
24:   if $U(S_c) > U(S)$ then
25:     $S$=$S_c$;// keep the global-best path to $S$
26:   end if
27: end while
28: return $S$.
```

A. Test case generation

The values of parameters considered in this paper are: $\alpha = 1$, $\beta = 2$, $q_0 = 0.9$, $\tau_0 = 0.1$, $CP = 0.1$, $\rho = 0.1$, $\xi = 0.1$, $W = [0.8, 0.2]$, $p1 = p2 = 0.5$, $noa = 10$, $MaxIt = 1000$. As the preferred parameter settings of the ACS algorithm for the TSP are given in [8], we use the same parameter settings. In Fig. 3 the ants will find $Path1$ and $Path2$.

$Path1 : \{AS_1, AS_2, AS_3, AS_4, AS_6, AS_7, AS_8, AS_9\}$
with a probability of p1=0.5;
$Path2 : \{AS_1, AS_2, AS_3, AS_5, AS_6, AS_7, AS_8, AS_9\}$
with a probability of p2=0.5.

In this paper, in order to find the solution with highest utility, the algorithm is designed to report the branch with the highest utility rather than the highest probability of execution. So, it will only returns one path with the highest utility.

The performance of the ant colony algorithm depends on the size of the data-intensive service composition problem. The size of the problem depends on the number of abstract services in the workflow, and the number of concrete services for each abstract service. Thus, we generate two test groups. The first test group includes 5 test scenarios with different numbers of service candidates. The number of service candidates for each abstract service ranges from 10 to 50, in increments of 10. The number of abstract services is fixed at 9. The second test group includes 4 test scenarios with different numbers of abstract services. The number of abstract services is 15, 20, 25 and 30. The number of service candidates for each abstract service is fixed at 10. This was done by increasing the number of abstract services on the workflow of Fig. 3 with a sequential flow structure. This two test groups are designed to test how the running time and iteration times of the proposed algorithm will change as the number of service candidates and the number of abstract services change.

All test scenarios are run twenty times and the average values are reported. The cost and response time of each service candidate are randomly drawn from a uniform distribution of the interval [1,10]. The access cost and transfer time of each data set are randomly drawn from a uniform distribution of the interval [1,10]. The transfer cost is the price to be paid per unit of transfer time and its value is set as 1 in this paper. The values of the access cost and transfer time of data sets, and the values of the QoS attribute cost and response time of service candidates will not affect the running time and iteration times of the proposed algorithm. So their values can be in any intervals.

B. *Result analysis*

The simulation results of the first test group are shown in Fig. 4 on the next page. In Fig. 4(a) to Fig. 4(e), the blue line denotes the utility of the best path from the beginning of the trial, and the red point denotes the utility of the best path of each iteration. The value of 'GUtility' is the utility of the best path from the beginning of the trial and it depends on the QoS attributes of services. Different values of QoS attributes give different values of 'GUtility'. That is to say, the change of the value of 'GUtility' has no significance for the simulation results. The value of 'FRIT' is the number of iterations when the best utility appeared and from this iteration its value will not change. According to the value of 'FRIT' in Fig. 4, as the number of concrete services increases, the ants need more iteration times to find the best path. Fig. 4(f) shows the effect of the number of service candidates on the running time for finding the best path. This figure shows the time consumption is basically linear to the number of service candidates.

The results of the second test group are shown in Fig. 5. According to the value of 'FRIT' in Fig. 5(a) to Fig. 5(d), as the number of abstract services increases, the ants need more iteration times to find the best path. Fig. 5(e) shows the effect of the number of abstract services on the running time for finding the best path. This figure shows the time consumption is basically linear to the number of abstract services.

VI. Related work

Quite a few studies have proposed using a workflow-based model to solve service composition problems. Bio-inspired algorithms are one type of the main approaches. One of our earlier studies [20] presented a survey on bio-inspired algorithms for Web service composition by comparing ant colony algorithms, genetic algorithms, evolutionary algorithms, and particle swarm optimization algorithms. In [22], [24], [26], [27], the authors applied an ACO algorithm to solve a Web service composition problem. However, these studies did not address the parallel, conditional, or loop execution of services which are very common in business processes or workflows. The paper [29] presented a QoS-aware Web services selection model using an AND/OR graph, but it did not have the ant replication and death rule to deal with AND vertices. The paper [9] used an AND/OR graph to represent a service dependency graph and presented its search algorithm for composite Web services. However, the authors focused on how to construct the graph to capture the input/output dependencies among the Web services, and they neglected the nonfunctional characteristics of services.

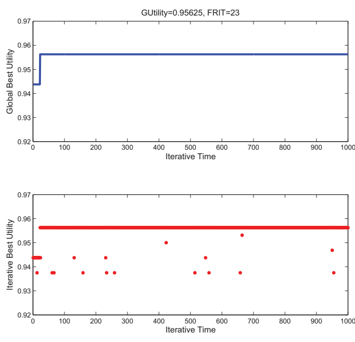
On the other hand, as we discussed earlier, data-intensive services have become the most challenging type of applications in service oriented computing. The data intensity and the communication cost of mass data transfer will affect the efficiency of data-intensive applications. The goal of our data-intensive service provision mechanism is to find the best providers for each elementary service and the best data-sourcing strategies for sourcing data which cannot only satisfy the service composer's QoS request, but also ensure the overall cost to use these elementary services and data is as low as possible, while service providers and data providers are still profiting from such an agreement. One of our earlier studies [21] presented a bio-inspired cost minimization model for data-intensive service provision. That paper was the first effort to address lower cost data-intensive service composition. We continued that work and presented an algorithm based on the ant colony system along with the evaluation of its performance through simulations in this paper.

VII. Conclusion

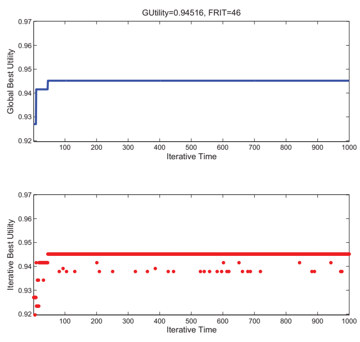
As explained in this paper, it is important to develop mechanisms for selecting concrete services and data replicas to achieve a cost-effective solution for data-intensive service composition. We are interested in how to apply bio-inspired approaches to optimizing the service cost, data cost, data transfer cost, and service-composition cost in data-intensive service provision. In this paper, the data-intensive service composition problem is modeled as an AND/OR graph, which is not only able to deal with sequence relations and switch relations, but is also able to deal with parallel relations between services. A novel service selection algorithm based on ant colony optimization is proposed, and the performance of the algorithm has been studied by simulations. Dynamic service-price-setting models using non-standard pricing mechanisms are expected to appear for composing data-intensive services. The design of the algorithms using negotiations for data-intensive service composition is currently under way.

Acknowledgment

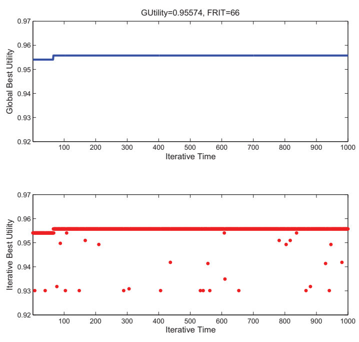
The authors wish to express their sincere gratitude to Dr. Madeleine Cincotta for her help in the editing of this manuscript.



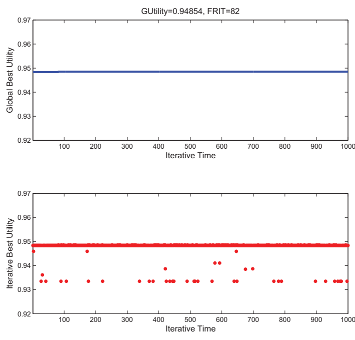
(a) Utility of each iterative course for 10 service candidates.



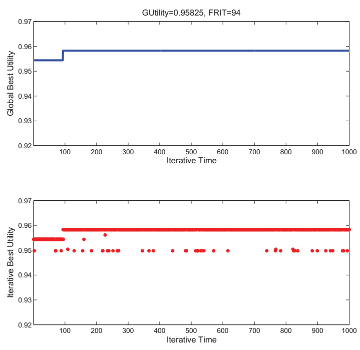
(b) Utility of each iterative course for 20 service candidates.



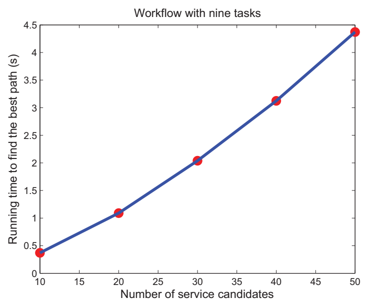
(c) Utility of each iterative course for 30 service candidates.



(d) Utility of each iterative course for 40 service candidates.



(e) Utility of each iterative course for 50 service candidates.



(f) The effect of the number of service candidates on the running time to find the best path.

Fig. 4: The results of the first test group

References

[1] A. Bucchiarone, and L. Presti, "QoS composition of services for data-intensive application", *International Conference on Internet and Web Applications and Services (ICIW' 07)*, pp. 46, 2007.

[2] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut, "Quality of service for workflows and Web service processes", *Web Semantics: Sciences, Services and Agents on the World Wide Web*, Vol. 1, No. 3, pp. 281-308, 2004.

[3] M. Carman, F. Zini, L. Serafini, and K. Stockinger, "Towards an Economy-Based Optimisation of File Access and Replication on a Data Grid", *Proceedings of 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 340-346, IEEE computer society, Washington, DC, USA, 2002.

[4] F. Casati, M. Sayal, and M. C. Shan, "Developing e-services for composing e-services", *Proceedings of 13th International Conference on Advanced Information Systems Engineering(CSiSE)*, Interlaken, Switzerland, Springer Verlag, pp. 171-186, 2001.

[5] Z. Chen, H. Wang, and P. Pan, "An approach to optimal Web service composition based on QoS and user preferences", *Proceedings of the 2009 International Joint conference on Artificial Intelligence*, pp. 96-101, 2009.

[6] A. Chervenak, E. Deelman, M. Livny, M. Su, R. Schuler, S. Bharathi, G. Mehta, and K. Vahi, "Data placement for scientific applications in distributed environments", *Proceedings of IEEE/ACM International Conference on Grid Computing*, pp. 267-274, 2007.

[7] M. Dorigo, and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem", *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 53-66, 1997.

[8] M. Dorigo, and T. Stutzle, "Ant colony optimization", MIT Press, Cambridge, MA, USA, 2004.

[9] Q. H. A. Liang, and S. Y. W. Su, "AND/OR Graph and Search Algorithm for Discovering Composite Web Services ", *International Journal of Web Services Research*, Vol. 2, No. 4, pp. 46-64, 2005.

[10] Y. Li, and C. Lin, "QoS-aware service composition for workflow-based data-intensive applications", *IEEE International Conference on Web Services*, pp. 452-459, 2011.

[11] B. Medjahed, A. Bouguettaya, and A. Elmagarmid, "Composing web services on the semantic web", *The VLDB Journal*, Vol. 12, No. 4, pp. 333-351, 2003.

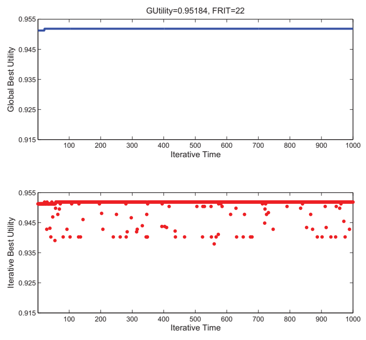
[12] M. Milenkovic, E. Castro-Leon, and J. R. Blakley, "Power-aware management in cloud data centers", *CloudCom*, LNCS 5931, pp. 668-673, 2009.

[13] C. B. Pop, et al., "Ant-inspired Technique for Automatic Web Service Composition and Selection", *12th International symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pp. 449-455, 2010.

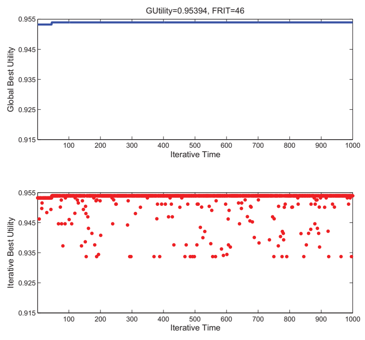
[14] J. Shen, G. Beydoun, S. Yuan, and G. Low, "Comparison of bio-inspired algorithms for peer selection in services composition", *IEEE International Conference on Services Computing (SCC 2011)*, pp. 250-257, 2011.

[15] J. Shen, and S. Yuan, "QoS-aware peer services selection using ant colony optimisation", *Business Information Systems Workshops*, pp. 362-374. 2009.

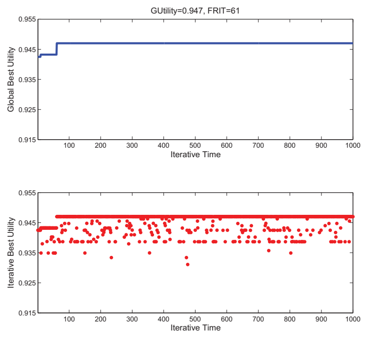
[16] S. N. Sivanandam, and S. N. Deepa, "Introduction to genetic algo-



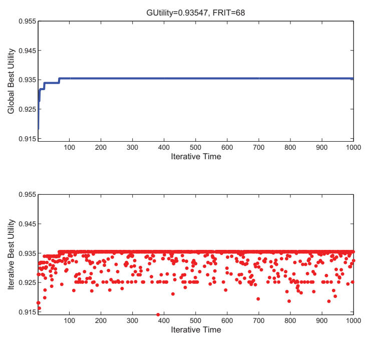
(a) Utility of each iterative course for 15 tasks.



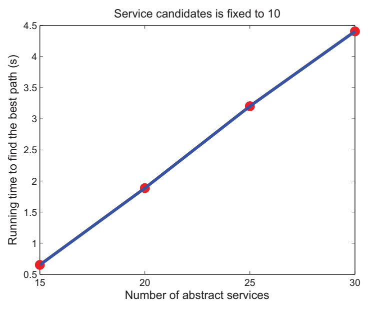
(b) Utility of each iterative course for 20 tasks.



(c) Utility of each iterative course for 25 tasks.



(d) Utility of each iterative course for 30 tasks.



(e) The effect of the number of abstract services on the running time for finding the best path.

Fig. 5: The results of the second test group

rithms", Springer-Verlag, Berlin, Heidelberg, 2007.

[17] Y. Song, H. Wang, Y. Li, B. Feng, and Y. Sun, "Multi-Tiered on demand resources scheduling for VM-based data center", *IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 148-155, 2009.

[18] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow Patterns", *Distributed and Parallel Databases*, Kluwer Academic Publishers, Vol. 14, No. 1, pp. 5-51, 2003.

[19] S. Venugopal, and R. Buyya, "An SCP-based heuristic approach for scheduling distributed data-intensive applications on global grids", *Journal of Parallel and Distributed Computing*, Vol. 68, No. 4, pp. 471-487, 2008.

[20] L. Wang, J. Shen, and J. Yong, "A Survey on Bio-Inspired Algorithms for Web Service Composition", *International Conference on Computer Supported Cooperative Work in Design*, pp. 569-574, 2012.

[21] L. Wang, and J. Shen, "Towards Bio-Inspired Cost Minimisation for Data-intensive Service Provision", *IEEE International Conference on Services Economics*, pp.16-23, 2012.

[22] R. Wang, L. Ma, and Y. Chen, "The Application of Ant Colony Algorithm in Web Service Selection", *International Conference on Computational Intelligence and Software Engineering*, pp. 1-4, 2010

[23] S. Wang, W. Shen, and Q. Hao, "Agent based workflow ontology for dynamic business process composition", *Proceedings of 9th International Conference on Computer Supported Cooperative Work in Design (CSCWD '05)*, pp. 452-457, Coventry, UK, 2005.

[24] Y. Wang, "Application of Chaos Ant Colony Algorithm in Web Service composition based on QoS",*International Forum on Information Technology and Applications*, pp. 225-227, 2009.

[25] M. Winter, "Data center consolidation: A step towards infrastructure clouds", *CloudCom*, LNCS 5931, pp. 190-199, Springer, Heidelberg, 2009.

[26] Y. Xia, J. Chen, and X. Meng, *On the dynamic ant colony algorithm optimization based on multi-pheromones*, *IEEE/ACIS International Conference on Computer and Information Science*, pp. 630-635, 2008.

[27] Z. Yang, C. Shang, Q. Liu, and C. Zhao, "A Dynamic Web Services Composition Algorithm Based on the Combination of Ant Colony Algorithm and Genetic Algorithm", *Journal of Computational Information Systems*, Vol. 6, No. 8, pp. 2617-2622, 2010.

[28] K. P. Yoon, and H. C. Land, "Multiple attribute decision making: an introduction", *Sage university papers series (quantitative applications in the social sciences*, Thousand Oaks, CA, Sage Publications, 1995.

[29] H. Yu, and M. Liu, "A QoS-Aware Web Services Selection Model Using ANDOR Graph", *Advanced Data Mining and Applications*, LNCS 7120, pp.124-137, 2011.

[30] S. Yuan, J. Shen, and A. Krishna, "Ant inspired scalable peer selection in ontology-based service composition", *IEEE World Congress on Services*, pp. 95-102, 2009.

[31] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, H. Chang, "QoS-aware middleware for Web services composition", *IEEE Transactions on Software Engineering*, Vol. 30, No. 5, pp. 311-327, 2004.

[32] Y. Zhang, X. Zhou, and Y. Gao, "Optimizing the data intensive mediator-based web services composition", in APWeb. Springer, LNCS 3841, pp. 708-713, 2006.

[33] Workshop on massive data analytics on scalable systems, at http://ceng.usc.edu/ simmhan/DataMASS2012/, accessed on Jan. 1st, 2013.